# Practical 5

~~Coast~~ Implement Conal Selection algorithm:

Theory:

1) Conal selection algorithm is optimization algorithm ~~inspired by the Conal selection theory~~ is a bio-inspired optimization algorithm based on principles of human immune system.
2) It mimics the behavior of B-cells (antibodies) which:
   - Recognize antigens (foriegn substances),
   - Clone themselves (proliferation),
   - Mutate slightly (to improve affinity),
   - Survive if they better recognize the antigen (natural selection).
3) Similarly ~~tot~~ CONALG tries to evolve better solutions in a computational problem-solving context.
4) It belongs to the class of Artificial Immune System (AIS) technique, which simulate the learning and memory capability of the biological immune system.
5) Conal selection algo is primarily designed to solve optimization ~~(function opti, param tuning)~~, pattern recognition ~~(chars, faces)~~ and classification problems by imitating how B-cells ~~it~~ evolve to recognise antigens.

5) Worbing:-

i) Initialization:
- Randomly generate a population of candidate solutions called antibodies.
- Each antibody is typically represented as a real-valued ~~&~~ or binary vector.

ii) Fitness Evaluation (Affinity Calculation):
- Evaluate how good each antibody is using fitness function.
- This measures how well the solution matches the problem.

iii) Selection:
- Select the top-performing antibodies with highest fitness scores.

(iv) Cloning:
- Produce multiple copies (clones) of selected solutions.
- The better the fitness, the more clones an ~~ant~~ solution might produce.

(v) Hypermutation: (some changes)
- Apply mutation to each clone with a certain probability.
- Mutation introduces diversity and explores nearby solutions, allowing the algorithm to escape local optima.

(vi) New population formation:
- Evaluate the fitness of the mutated clones.

- From the combined set of original solutions and mutated clones select the best individual to form the next generation.

(vii) Iteration:-
- Repeat steps 2 - 6 for a predefined number of iterations until convergence criteria are met.

(viii) Result:
- The best solution after all iterations is returned as the final optimal solution.

7) Key concepts:
- Antigen - problem to be solved.
- Antibody - solution to the problem.
- Affinity / fitness - How good the solution is.
- Cloning - replicating good solutions to give them more chances.
- Hypermutation - Random mutations to clones to discover better solutions.
- Selection - keeping only the best solutions for next generations.

8) Applications:-
① Optimization:- function optimization, parameter tuning.
② Pattern recognition:- Character recognition, facial recognition.
③ Anomaly detection:- fraud detection, network intrusion detection.

(4) Robotics :- path planning.

9) Clonning Process :-

- After we select the top antibodies, each selected antibody is copied (clone factor times).

10) Mutation Process :-

- Mutation means making a small ~~rand~~ random changes to the clones.
- After cloning we slightly add some changes to each value (gene).
- Mutation happens with a probability eg: if a clone A has gene 0.6, mutation ~~maps~~ may slightly change it to 0.65 or 0.55.

11) Selection Process :-

- Selection means keeping only the best solutions after mutations.
- After mutation:
  we have old population + new mutated clones (large number).
- We evaluate all individuals based on fitness.
- We then select only the top ones ~~ato~~ (based on fitness) to survie for the next generation.

12) Fitness :-

- Fitness is a numerical value that tells us how good a particular solution is.
- It's ~~selected~~ calculated by a fitness function, higher fitness → better solution.

· Fitness function in code:

```
def fitness_function(x):
    return(-np.sum((x-0.5)**2)
```

· x is an antibody (solution vector like [0.2, 0.7, 0.4, 0.8, 0.5])
· We ~~calc~~ calculate how close each value ~~is~~ in x is to 0.5.
· We square the distance from 0.5 → $(x-0.5)^2$
· Then we sum al these squared distanses → sum((x-0.5)^2).
· Then we negate the result → -np.sum((...))
· Therefor antibodiys closer to 0.5 get higher fitness.
· Goal is to minimize the distance to 0.5.

☆ Why negative sum?
· Normally, smaller distance means better solutions.
· But ~~to~~ CLONALG expects higher fitness = better.
· So we take negative of the distance, (smaller distance become higher fitness values).

☆ Why 0.5?
In real-world problem, the target could be anything — 0.5 is just a simple, symmetric point in the range [0-1]